1-1-2004

Integrating mobile computing with fixed network

Bing Zou
Iowa State University

Integrating mobile computing with fixed network

by

Bing Zou

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Leslie Miller, Major Professor
Sarah Nusser
Shashi Gadia

Iowa State University

Ames, Iowa

2004

Graduate College
Iowa State University

This is to certify that the master's thesis of

Bing Zou

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

Abstract

With the recent developments in science and technology, mobile computers are getting lighter, more powerful and even cheaper. More and more government agencies are using mobile computers to collect and use geospatial data in the field operations for federal agencies and industries. In order to integrate heterogeneous geospatial data resources to field mobile computing environment, such as survey and other types of data collection, in our research, an infrastructure is designed and implemented. Additionally, to integrate mobile computing with the fixed network, field wrapper is introduced to the infrastructure. A detailed analysis and design of the field wrapper is discussed, also a UML design of the field wrapper is presented. In implementation level, in order to address the current needs in field mobile computing and unsolved issues, for example, the firewall and security issue, some new technologies are leveraged and applied such as SOAP, MVC design pattern, code generation, etc. Mobile collaboration is also discussed along with implementation suggestions.

Chapter 1 Introduction and Background

1.1 Introduction

With the recent developments in science and technology, people now are able to use computers as small as their palms to collect and analyze data, reply to emails, communicate with the other devices etc. These palm size computers (a.k.a. mobile computers) are playing a more and more important role, especially in field. More and more government agencies are collecting and using geospatial data in the field operations for federal agencies and industries. For example, surveyors can use mobile computer to find a sample point with the integrated GPS device and software, collect data, collaborate with experts or their supervisors to analyze the data, and determine whether more work is needed or to calculate the next sample point, etc. In this kind of scenario, mobile computers must be able to retrieve geospatial data from a variety of geospatial databases, must be able to connect with other field staff and must be able to communicate with the supervisors or experts all over the world. Therefore, the mobile computers must be integrated with a fixed network to be able to communicate with other mobile computers, laptops, desktops or resources on the Internet.

Although there are more and more mobile computers being integrated with wireless cards or with an interface to connect with a wireless card, there are still some unsolved problems:

1. In most cases, there is no access point in the field for the mobile computers to connect with. In another word, it is difficult for mobile computers to directly connect with a fixed network.
2. A lot of existing mobile computers are not powerful enough to perform relatively complicated computing tasks. Also because the mobile computers suffer from the lack of storage, memory and small screen size, geospatial data tend to be very large and need to be transferred via the relatively slow wireless networks.

3. Field personnel who are using the mobile computers are free to roam from one place to another [4], hence the mobile computers might leave one wireless network and enter another one from time to time, or the mobile computers might be disconnected from the network for a while and come back later. How to resume the state of the mobile computers in these scenarios is another issue.
4. Security is always a big issue. As one of the main methods of providing security, a firewall would also cause some difficulty in connecting two computers in different networks.
5. Mobile collaboration. Mobile computers represent one of the fastest growing segments of the computer industry today. As this happens, they become a natural medium for collaboration. Besides storing much of the user's personal data, they are always at hand, in sharp contrast with desktop computers. However, much work remains to be done for mobile computers to become the platform of choice for computer-mediated collaboration [5].

The contribution of this thesis lies in several aspects. First, in order to integrate the mobile computers with the fixed network, a field wrapper is introduced, modeled and implemented to make the mobile computers seamless integrated with the fixed network. Second, the SOAP technology is introduced and implemented to solve the firewall issue. Finally, the scenarios of collaboration are discussed and the solutions to the issues are presented.

This thesis is organized as follows. Chapter 1 gives brief introduction and background, after that, the model as well as the UML design of the field wrapper are discussed in Chapter 2. Chapter 3 is about the implementation of the field wrapper. The SOAP technology and implementation is also covered in Chapter 3. The discussion about mobile collaboration is in Chapter 4. Finally, Chapter 5 draws conclusions and proposes future works.

1.2 Needs in mobile computing

With the development of mobile computing technologies, there is a fast growing trend in using mobile computing in all kinds of field operations, especially in surveying.

In mobile field data collection, a data gatherer requires information services, especially geospatial information service, for support in navigating to the correct locations, describing context of observed objects and providing thematic information for sampling or analysis. While geospatial data is important to most of these applications, relatively little geospatial data is currently incorporated into them.

Research in mobile applications has focused on handheld and tablet devices using design principles developed for desktop telephone interviewing laboratories. As such, these endeavors have failed to anticipate emerging technologies such as augmented vision and wearable computers, infrastructures that facilitate more flexible and user-friendly interactions, and the exploitation of digital information resources such as geospatial data and other information beyond that currently supplied as part of a central survey database.

1.3 Current Mobile Computing and Supportive Technologies

The mobile computing environment can be described by the following attributes (a) mobile users, (b) mobile support stations or base stations serving an area, (c) wireless interface, (d) wireless medium with varying channel characteristics (due to fading, noise, interference, etc.) and (e) various applications requiring specific support.

A mobile computing environment raises such issues as how to route packets as the mobile user (hosts) moves from one place to the other. The mobile computing environment also presents challenges in terms of the amount of bandwidth available to the mobile users. Mobile users (as compared to fixed network users) typically have to work with a limited and variable amount of bandwidth due to the nature of wireless environment. The amount of bandwidth available to a user is also location dependent in many ways. For example, if a

mobile user moves to a location with many active users, then the user may receive little or no bandwidth at all. If the mobile user moves to a location where the amount of interference is high, then effective bandwidth to the user may also reduce substantially [4].

Two configurations have been proposed for mobile networks: "Ad-Hoc Configuration" and "Infrastructure-based Configuration." In the Ad-Hoc Configuration, the topology of the wireless network is not fixed because base stations are portable and can be moved around as needed. Communication is possible between mobile users by the cooperation of these portable base stations. In Infrastructure-based Configuration, the topology of the network is fixed and so are the base stations. Both these configurations have their pros and cons. Ad-Hoc configuration provides greater flexibility for some applications (say, troop movement) at the expense of greater overhead. This overhead stems from the use of broadcasting (or paging) that is typically employed to locate mobile users. It also limits the scalability of pure Ad-Hoc networks. Infrastructure-based configurations have been employed in cellular and Personal Communications Systems (PCS) networks and may use a centralized switch (or mobile switching center) for interconnecting fixed base stations, reducing the overhead involved in location management and updating.

Wireless communication technologies, such as IEEE 802.11 WLAN (Wireless Local Area Network) technology, CDPD (Cellular Digital Packet Data) and the satellite-based Iridium system, support a range of modes of interaction independent of traditional ties to the fixed locations of copper wire and fiber. New portable devices (e.g. personal digital assistants, handheld devices and wearable computers), which use more powerful CPU and have much more memory inside, are now approaching the ruggedness and specialized modalities required for field use. Significant recent developments in field technologies include cellular communications, highly portable miniaturized computing systems (e.g. wearable computer), special purpose input and output devices such as glasses and condensed keyboards, and voice synthesis and recognition software, referred to as VISIO (Voice-In, Sound and Images Out) interfaces. A wearable computer can offer two great field use

advantages. First, the system can present a field user with information in real time. When this information is placed into the vision field, it is a form of augmented reality. Secondly, the system can take advantage of distributed networks to retrieve map and image data for display based on the position of the field user [3].

Infrastructure components are also rapidly advancing to provide the data richness to support unstructured data. For example, the object-oriented paradigm is increasingly adopted as the basis for designing information systems. In addition to providing the ability to process complex data types, the object-oriented paradigm facilitates a modular system design, thereby increasing the level of code reuse in complex and dynamic applications. The use of data warehouses is also rapidly expanding. Storing both the tools and data in the same system has proven to be very valuable for both businesses and scientific applications. Mobile agents are generating new opportunities as well. Mobile agent systems augment the benefits of the client-server model with flexibility in computational load balancing and reductions in information transferred over the network. The security concerns generated by this flexibility are currently being addressed. When supplemented by more traditional connections (e.g., CORBA), mobile agent systems can provide a safe and effective means of rapidly developing complex distributed systems. Finally, mediators have proven extremely useful in systems designed to integrate heterogeneous data sources.

1.4 Current Approaches

In the 1970s, computer-assisted data collection systems were developed for sample survey applications (referred to as computer-assisted survey information collection, or CASIC). In its original conceptualization, CASIC software was intended to provide a tool for key entry and questionnaire flow control during the telephone interviewing process, using a suite of stand-alone desktop PCs in an office environment. As computer technologies evolved, CASIC system design migrated to a client-server model in the office environment, and laptops enabled computer-assisted face-to-face interviews in households. As an example

of an advanced computer-assisted data collection system, consider the CASIC system developed by the USDA Natural Resources Conservation Service and Iowa State University to conduct the NRI (National Resources Inventory) in mobile office and field settings [7]. A client-server architecture was developed in 1996 to support handheld computer-assisted data collection. The system consisted of:

- Redundant central database servers with RAID storage,
- An Oracle database containing tracking variables and historical and newly collected data for each area segment,
- Redundant front-end servers to negotiate database requests,
- About 500 handheld computers (Apple Newtons) equipped with a computer-assisted data collection form for recording values for each variable, and
- TCP/IP-based communication via a variety of modes (wire-line, wireless, local area network, and Internet) between the front-end server and the handheld client.

A data gatherer logs on to a front-end server for a short period to query the database and request samples to be worked on. During data collection, the client software notifies the user of apparent inconsistencies among collected variables, as well as with historical data stored onboard. These problems are resolved and the data gatherer returns the sample data to the server. Except for interfaces used to define queries and samples to be returned, most of the negotiation between the handheld computer and server system is hidden from the data gatherer.

The NRI CASIC system has provided significant improvements over the former stand-alone desktop systems. It has introduced small and large-scale mobility into the data collection environment, has improved data quality by effectively incorporating editing into the data gathering process, and has greatly reduced the time required to resolve problems during post-data collection processing, when compared with past efforts. However, a number of limitations are presented by this paradigm, as noted below:

Several components of this survey are explicitly geospatial, yet there is no mechanism to utilize digital geospatial data in the field. Currently, a mixture of analog and digital materials is used to collect data and it is difficult to integrate these sources in their current forms. The quality of data gathered and the efficiency of the various components of the process can be greatly improved by developing a framework that facilitates integration of digital geospatial data into these processes.

Structured relational data formats for input, storage, and queries have provided the utilities needed to collect data according to prescribed scientific protocols. However, the use of these tools requires special code to be developed to accomplish these functions. They lack the flexibility to incorporate geographic data and do not take advantage of integrated information system tools.

The client-server model requires an open connection in order to exchange information, limiting transfers to relatively small requests. Software agents could be used to send complex queries or request analyses that exceed the shorter connection times appropriate for the current system, returning results of the request at a later time.

The current system is designed for a single purpose. Hardware and some code are reused as new surveys are deployed. However, the system is not inherently modular or readily adaptable for new data collection efforts or interaction with activities in other parts of the agency or in other agencies.

While the client-server CASIC model has served the data collection process well, to fully exploit the potential of digital geospatial data and emerging information technologies, a new paradigm is needed for mobile field data collection.

1.5 General Infrastructure

The essence of the current infrastructure is to integrate the geospatial data sources. The model of the infrastructure is shown in Figure 1-1, which schematically describes the pathways of data exchange. A key feature of any environment designed to give field workers

access to geospatial data is the infrastructure used to connect the field devices to the data sources.

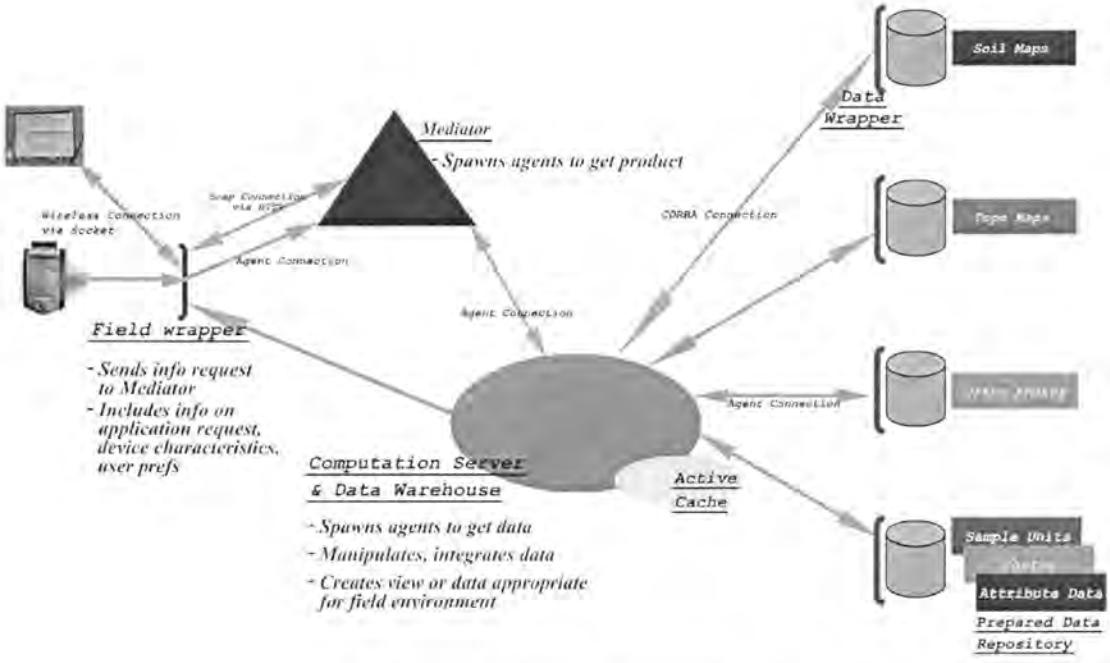

Figure 1-1. Current Infrastructure

As shown in Figure 1-1, the infrastructure consists of at least 6 components, which are Field User, Field Wrapper, Mediator, Computation Server, Data Wrapper and Data Sources.

- Field User. Field User represents all kinds of computing devices including Pocket PCs, Palms, Smart Phones, Tablets or Laptops.
- Field Wrapper (FW). Field Wrapper is an extension of existing wrapper concepts to support mobile computing environments. It acts as a proxy between the Field User and the Infrastructure, which manages the data translation and transmission.
- Mediator. Mediator is a class of software modules that mediate between the field device/wrapper and the data sources. It uses supporting information such as metadata and object-oriented views to generate a query string according to the predefined rules.

- Computation Server. Computation Server takes care of all kinds of computation tasks that could not be handled by the mobile computers or the wrappers, also it provide additional tool set for special computational usage.
- Data Wrapper. Data Wrapper is a wrapper that encapsulates the data sources. It localizes the details associated with heterogeneity in data sources.
- Data Sources. Data Sources represent all kinds of data sources, including data warehouse, government geospatial database and etc..

The following scenario demonstrates a typical data flow (working mechanism) of the infrastructure. A surveyor is sent to the field to do some sampling along a river, in order to locate the sample points, he sends a request about the hydrologic data and topographic information of the working field using the pocket pc. The request is first sent to a local Field Wrapper via wireless network. The Field Wrapper receives the request, authenticates the user's registration data, reconstructs the request according to predefined rules and sends it to the Mediator. The Mediator examines the request and finds out that two current running data sources have the requested data. The Mediator rebuilds the request and divides it into two sub-queries because the topographic data and hydrologic data locate in different data sources. Then the Mediator sends the request to the Computation Server. The Computation Server sends the two sub-queries to corresponding Data Sources. The Data Wrappers of the two Data Sources receive the query respectively; then they translate the queries to be executed by the wrapped Data Sources. When the Computation Server receives the results from the Data Sources via the Data Wrappers, it generates a new image that combines the two results and shrinks the image size to be fit in the LCD screen of the mobile computer. The Field Wrapper sends the result to the Field User when it receives the returned image, and finally the field user views the integrated map on his/her mobile computer.

Chapter 2 Design model of the Field Wrapper

2.1 Introduction

2.1.1 The Wrapper Technology

Generally speaking, a wrapper is a program that acts as an interface between a caller requesting an action and a wrapped component that is needed to execute the action. Its purpose is to create interoperability between the caller and the wrapped component, which may exist in environments that differ sufficiently to require some kind of translation of the request from caller to source and/or the results provided by the source to the user. Usually, the wrapper encapsulates the low level details of the wrapped component and provides high-level information about the wrapped component.

The model of a wrapper-based system is shown in Figure 2-1. When a request from a caller to the wrapped component reaches the wrapper, the wrapper will translate it into the format that the wrapped component can understand and process. Similarly, the wrapper will also translate the results returned from the wrapped component to the format that the caller wants.

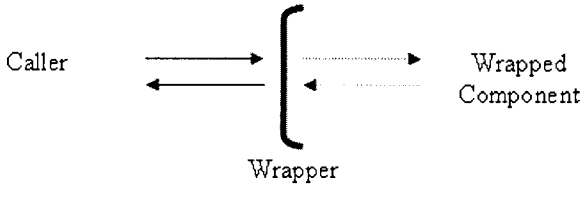

Figure 2-1. Model of wrapper based system

In general, a wrapper provides an interface between the user or caller's environment and the wrapped component. By adding such a wrapper, heterogeneity of different computing platforms can be overcome.

2.1.2 What is the Field Wrapper?

In this project, we extend the concept of wrapper to field data collection environments as a means of isolating the heterogeneity of field users, applications, and computing environments from infrastructure functions.

From the Field User's point of view, the field user running on PDA, handheld PC or tablet etc. is the caller and the whole infrastructure is the wrapped component in this case, and vice versa which is shown in Figure 2-2.

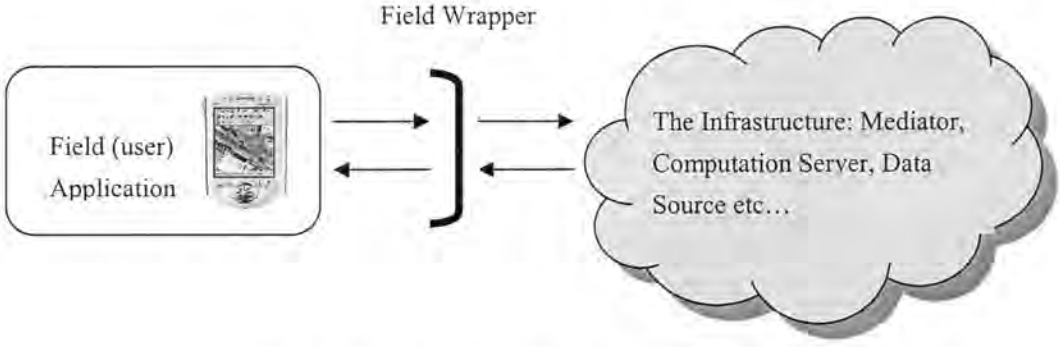

Figure 2-2. Concept Model of Field Wrapper

2.1.3 Why do we need a Field Wrapper?

Because of the need to use and collect geospatial information in a potentially limited mobile computing environment, the goal is to enable any device with any user in the field to readily and easily interact with a larger computing infrastructure to search, retrieve, or return geospatial data along with other kinds of data. As the interface between the field user and the infrastructure, it's obvious that major function of the field wrapper is to provide seamless integration between the field user and the infrastructure. As we will see in the following section, because of the limitation of the mobile computing environment, we need the field wrapper to be powerful enough to provide reliable service and efficient enough to provide fast service for the field user.

2.2 Functions of the Field Wrapper

Now we know that the major role of the field wrapper is a communication and translation node between the field user and the infrastructure. The next step is to analyze the functions and characteristics of the field wrapper in detail to provide a theoretical basis for the implementation section. To do this, we should analyze and divide the needs of the user in different cases to determine how to provide different functions in corresponding cases. After that, an emphasis of the characteristics of the field wrapped will be presented on behalf of the field user, which will have significant impact on the final implementation of the field wrapper in addition to the functions it provides.

2.2.1 Need of the user (Field data collection)

Roughly speaking, the field tasks in data collection can be divided into three aspects as followed:

- Planning
  - Creating routes, navigation path, workload sequence
  - Geospatial data needs: road maps, photographs, topographic maps
- Navigation
  - Moving by car or other ways to each data collection site
  - Geospatial data needs: integrate GPS signal with maps
- Data collection
  - Involves focused attention, limited local movement on foot
  - Geospatial data needs: add thematic data to navigation resources

In order to make the designation and implementation of the field wrapper easier, we should analyze the need of the user in some different points of views, so that it will be much clearer for us to determine what kind of service or function should be provided by the field wrapper in which case.

2.2.1.1 Different computational capability of the field device

As far as the computational capability of the field device is concerned, we could divide the needs of the user into at least two cases:

1. **General request**

As far as we know, most of the requests from the user will be the request of a certain type of maps of a certain area. All these kinds of requests will go to the mediator for determining the most suitable data sources, locating the corresponding data source and generating the appropriate query, which is called "mediation".

2. **Request of additional computation**

There are hundreds of types of mobile devices available for the field users nowadays such as pocket pc, smart phone, tablet and so on. Some of them only have very limited capability of computation, where the application has to request a modified map from the infrastructure even when the user wants to zoom in/out on the current map.

2.2.1.2 Mobility concern

As far as the mobility of the user is concerned, we can divide the needs of the user into two cases:

1. **Moving**

The user is moving by car or on foot or by some other facilities from one place to another, the user needs the map change simultaneously or frequently as his movement to indicate the direction, route or path he is taking or should take. One particular example is just like a GPS device.

2. **Non-mobile**

In this case, the user is not moving or the map is unrelated to the movement of the user.

2.2.1.3 Different operations to the data sources

As far as the modification of the data sources is concerned, we can divide the needs of the user into another set of two cases:

1. **Read Only data sources**

In some cases, the user only retrieve data from the infrastructure and will not modify the data, for instance, topographic map, road map, etc..

2. **Writable data sources**

In some cases, the user can not only read from a particular data source, but he/she can write new or modified data/map back to that data source. For example, a field worker who is doing a census or some investigation or data collection will need to send those data back to the data source (server).

2.2.1.4 User capability

Those who are handicapped still have the opportunity to use the mobile devices. Of course we should provide some special aids for those people to make use of the services provided by the infrastructure. Some of the solutions developed for handicapped individuals can be generally useful due to special circumstances, for example, a person driving in bumper to bumper traffic on the highway can not make use of a map on the small screen.

2.2.2 Required functions of the field wrapper

With the analysis of the needs, we can separate these functions into two sections: general functions and solutions for specific cases.

2.2.2.1 General functions

First of all, the field wrapper shall provide the most general functions as a wrapper. In additionally, nowadays the security issue is becoming more and more important and so is the reliability concern. According to these considerations, the general functions of the field wrapper could be organized as followed:

- **Providing the connection between the field user and the infrastructure**

  The field wrapper shall be able to receive the request from the field user, deliver the request to the appropriate component in the infrastructure, receive the returning result

from the infrastructure and transfer it back to the requesting application. In additional to these basic connection functions, the field wrapper shall provide more functions such as:

- **User identification and registration (Log in/out)**

  The field wrapper shall be able to verify the user information of any people who want to use the infrastructure so that the user can make use of the services according to his user id and password.

- **Reliable transportation**

  The field wrapper shall be able to guarantee that the field user can receive and only could receive the returned files based on his/her previous requests. And in some particular cases such as transferring a sequence of files to the field user, the field wrapper shall be able to check whether the files are received correctly and in the right order.

- **Request/Result translation**

  As the interface between the field user and the infrastructure, the field wrapper shall be able to parse the request from the field user and send it to the appropriate component in the infrastructure. While receiving a result from the infrastructure, the field wrapper shall be able to determine the delivery mode, the destination user, etc..

2.2.2.2 Solutions (more functions) for the specific cases

We have analyzed the need of the user in different cases in the previous section. From those analyses, we know that according to the different specific needs of the field user, more functions shall be provided by the field wrapper. The discussion of solutions to the different situations is processed case by case as followed:

- **Different computational capability of the field device**

  According to the different computation capabilities of different devices, at least two types of requests, general requests and requests of additional computation can be made by the field user. For the requests of additional computation, the current solution is to send this

kind of request directly to the Computation Server where the data warehouse is located and the previous results are stored in the local cache so that there is no need to mediate the request when the same data is needed.

Therefore the field wrapper must be able to parse the request and find out whether the request is an additional computation request, and then if so, the field wrapper shall be able to send the request to the Computation Server directly.

In fact, another possibility exists for additional computation requests. It is possible that some computational tasks could not be accomplished by a mobile device even it has a high speed CPU (for example, Significant delay), or some commercial third party tools such as some statistic computation tools are only available in the server side. We will encounter the same situation when a light client requests additional computation.

- **Mobility concern**

  A special case comes about when the user is moving and he/she wants the map to change simultaneously with his movement. The map must be shown with enough detail and clarity yet support frequently change.

  The key issue here is how to predict which map will be needed next. A possible solution is that the application use GPS to track the movement and direction of the user, then it could generate and send a request of a larger area map according to the current setting and moving direction.

  The possible function of the field wrapper in this case is that while a light client is moving, the field wrapper shall be able to act as a cache for the user to ensure the timeliness.

- **Different operations to the data sources**

  The possibility of writing data back to the data sources only occur in a few specific data sources and most of the data sources are read-only. The solution to this case is to link the specific data source and the user directly through the field wrapper (read and write), which means we don't need the mediator to perform mediation because the users should

know what and where the data source is and they have proper user id and password to access.

- **User capability**

  Although the user capability is another issue, it should be accommodated by the application interface design.

2.2.2.3 Discussion of performing mediation

If the mediator deals with all the mediation tasks, it will be much easier to implement the field wrapper because all the requests from the user will be sent to the mediator directly no matter what kind of request it is.

But if the efficiency and the promptness issues are taken into account, the field wrapper needs to act as the first mediation gate. This is a better solution than leaving all the mediation tasks to the mediator. Nobody wants to wait a minute for a returned result; promptness is always one of the most important things that the user cares about. If we leave all the mediation tasks to the mediator, it is no doubt that the mediator will be loaded with a serious burden and the mediator will become the bottleneck of the whole system, then the efficiency of system will be very low and the promptness demand of the user will not be met.

In a word, to give the field wrapper some basic mediation functions such as directly sending requests to some specific data sources or the Computation Server will balance the high burden of the mediator and increase the efficiency and promptness of the infrastructure.

2.2.2.4 Expanded diagram based on the functions of the field wrapper

The expanded system structure diagram is shown in Figure 2-3.

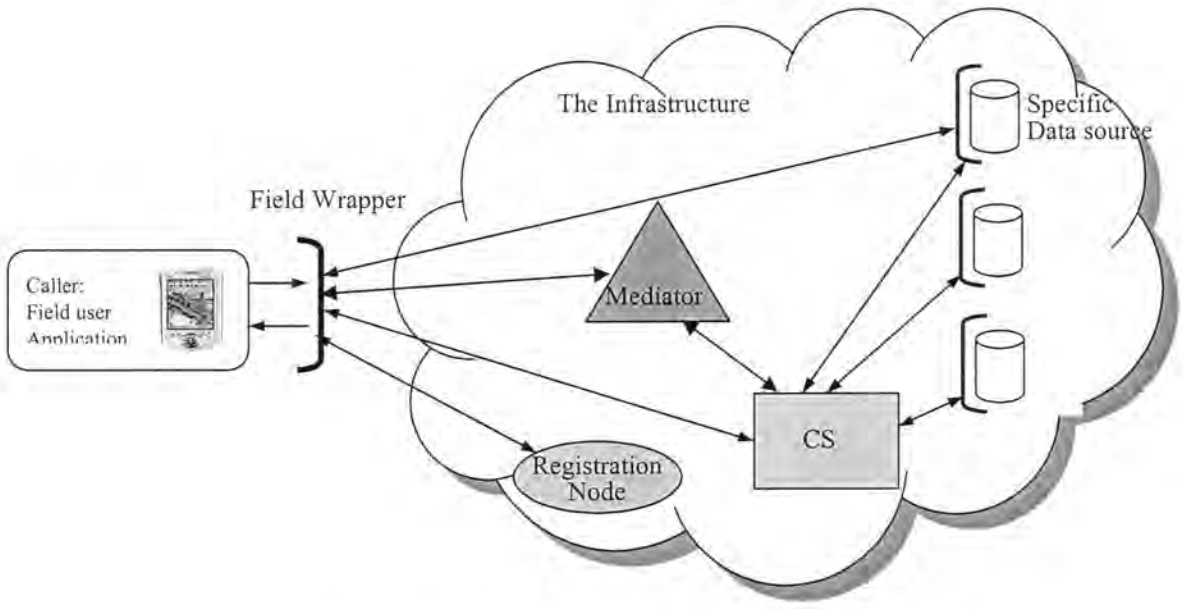

Figure 2-3. The Expanded System Diagram

2.3 Characteristics of the Field Wrapper

The following characteristics should be taken into account before implementation.

- **Compact**

  Although we want the field wrapper to have some basic mediation functionality, we won't replace the mediator with it. A wise decision is to leave most of the mediation work to the mediator, and focus on the efficiency and reliability issues in the field wrapper.

- **Efficient**

  A field user always wants the result to be returned as prompt as possible. So the key issue of the field wrapper is how to make it efficient enough to provide a prompt response for the user.

- **Reliable**

The field environment is so complicated that the reliability of the services provided by the infrastructure is very important especially in this case. For example, if the device lost the connection while waiting for the result of the submitted request, the field wrapper shall be able to confirm the delivery or keep the returned result and continue trying until the delivery is confirmed.

- **Secure**

  We only want authorized users to use the system. For example, some government databases could only be accessed by authorized users. As the gate to the whole infrastructure, the field wrapper must be able to verify the user id before the user sends any requests to the Infrastructure. While some data, for example, government data, requires data confidentiality, an encrypted connection is also required.

- **Easy of use**

  In order to test and demo the field wrapper, and manage the field wrapper more conveniently, the field wrapper shall be easy for the administrators to use. For example, a simple console or GUI could be handy in this case.

- **Expandable / Maintainable**

  New functions shall be able to be added to the field wrapper easily, which requires the code of the field wrapper to be well designed and with high flexibility. Also coding standards such as comments to make the code readable and maintainable must be used.

- **Easy for assembling**

  Different versions of the field wrapper will be needed to provide different functions in addition to or different from the general functions. The object-oriented paradigm is used to make the field wrapper like a machine assembled by different components (objects).

2.4 UML Design of the Field Wrapper

2.4.1 Introduction of UML

The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. UML is not a software process model or a system development methodology; it is a notation, a mechanism to "pen the problem" in such a way as to uncover the essence of an application domain. The goal of UML is to provide a consistent model for proper software implementation. All of the artifacts that UML delivers are traceable. With UML, the project will not only produce less useless and "go-nowhere" deliverables, but it will serve as a checkpoint of the previous model's soundness. Because the UML models are interlocked in their creation, identifying when a component is missing or potentially incorrect is easier.

The primary design goals of the UML are as follows:

- Provide users with a ready-to-use, expressive visual modeling language to develop and exchange meaningful models.
- Furnish extensibility and specialization mechanisms to extend the core concepts.
- Support specifications that are independent of particular programming languages and development processes.
- Support higher-level development concepts such as components, collaborations, frameworks and patterns.

2.4.1.1 Diagrams of UML

UML consists of nine different, interlocking diagrams of a system:

1. Activity
2. Class
3. Collaboration
4. Component

5. Deployment
6. Object
7. Sequence
8. State
9. Use-case

And the package diagram is also an important diagram in UML, but not one of the nine key diagrams. It was formerly called the subsystem diagram in other notations and can contain any and all of the diagrams listed above.

In UML, a scenario is a sequence of steps describing an interaction between a user and a system. A use case, then, is a set of scenarios tied together by a common user goal. Often, a use case has a common all-goes-well case, and many alternatives that may include things going wrong and also alternative ways that things go well. An actor is a role that a user plays with respect to the system. Actors carry out use cases. A single actor may perform many use cases; conversely, a use case may have several actors performing it. Actors don't need to be human. An actor can also be an external system that needs some information from the current system.

2.4.1.2 Relationships between UML and Java

The java notations corresponding to the UML notations are listed in Table 2-1.

Table 2-1. Relationships between UML and Java.

UML Diagram	Specific Element	Java Counterpart
Package	Instance of	Java packages
Use-case	Instance of	User interface artifacts (downplayed early on) in the form of pathways that will eventually become sequence diagrams
Class	Operations	Operations/methods
	Attributes	Member variables and related access or operations
	Associations	Member variables and related access or operations
Sequence	Instance of	Operations in a controller class to coordinate flow
	Message target	Operation in the target class
Collaboration	Instance of	Operations in a controller class to coordinate flow
	Message target	Operation in the target class
State	Action/activities	Operations in the class being life cycled

Table 2-1. (Continued)

UML Diagram	Specific Element	Java Counterpart
	Events	Operations in the class being life cycled or in another collaborating class
	State variables	Attributes in the class being life cycled
Activity	Action states	Method code to implement a complex operation or to coordinate the messaging of a use-case pathway
Component	Components	Typically one .java and/or one .class file
Deployment	Nodes	Physical, deployable install sets destined for client and/or server hosting

2.4.2 UML Analysis and Design

2.4.2.1 Functional Decomposition of the Field Wrapper

From the analysis of the functions and characteristics of the field wrapper, we could decompose the field wrapper into at least six components:

RequestServer

RequestProcessor

RequestSender

ResultServer

ResultProcessor

ResultSender

Some other components such as the User Manager and the GUI are optional here. The details will be discussed in Chapter 3.

2.4.2.2 Actor Identification

To identify the actors in the field wrapper subsystem, we should proceed by asking 4 questions:

Who/what will be interested in the system?

FieldUser / Infrastructure

Who/what will want to change the data in the system?

RequestProcessor / RequestServer / ResultProcessor / ResultSender

Who/what will want to interface with the system?

RequestServer / ResultServer

Who/what will want information from the system?

FWAdmin

Proposed actors from of the field wrapper sub-system are listed in Table 2-2.

Table 2-2. Proposed Actors for Field Wrapper.

Actor	Definition
FieldUser	Submits requests to the Field Wrapper.
Infrastructure	The other parts of the infrastructure that parses the requests.
RequestServer	Listens to the requests from FieldUser.
ResultServer	Listens to the results from the Infrastructure.
RequestProcessor	Translates requests from the FieldUser to a format that the Infrastructure can understand.
ResultProcessor	Translates results from the Infrastructure to a format that the FieldUser can understand.
RequestSender	Sends processed request to the Infrastructure.
ResultSender	Sends translated result back to caller FieldUser.
FWAdmin	Requests reports from the system.

2.4.2.3 Use-case Identification

After identifying the actors, the use-cases could be identified as in Table 2-3.

Table 2-3. Proposed Use-case List.

Actor	Use-case	Included Events
RequestServer	Receive Requests	FieldUser submits/cancels request. RequestServer receives request.
RequestProcessor	Process Requests	RequestProcessor processes request.
RequestSender	Send Requests	RequestSender sends request.
Infrastructure	Processes Requests	Infrastructure processes request and send back results. (Not in the scope of the field wrapper)
ResultServer	Receives Result	ResultServer receives result.
ResultProcessor	Process Results	ResultProcessor processes result.
ResultSender	Send Results	ResultSender sends result.
FWAdmin	Manage FW	FWAdmin performs a maintaining task.

With the use-cases, the next step is to find the pathways through them. Three levels of pathways – primary, alternate, and exception are identified as listed in Tables 2-4, 2-5 and 2-6.

Table 2-4. Happy Paths for the use-case.

Use-Case	Happy Path
Receive Requests	RequestServer receives requests from valid FieldUser.
Process Requests	A regular request from FieldUser is translated correctly.
Send Requests	A request is sent to the Infrastructure and is received successfully.
Receive Results	A result is sent by the Infrastructure and received by the ResultServer successfully.
Process Results	A result form Infrastructure is translated correctly.
Send Results	A result is sent back and received by the caller (FieldUser) successfully.
Manage FW	FWAdmin performs a maintaining task and succeeds.

Table 2-5. Alternate Pathways for the use-case.

Use-Case	Alternate Path
Receive Requests	None.
Process Requests	The FieldUser provides the correct format that need not to be retranslated.
	An ID verification (login) request is received.
	A logout request is received.
	A Cancel request is received.
Send Requests	None.
Receive Results	None.
Process Results	A result for an ID verification request is received.
	A result for a cancelled request or logged out FieldUser is received.
Send Results	FieldUser comes back alive and an "Alive" signal is received.
Manage FW	None.

Table 2-6. Exception Pathways for the use-case.

Use-Case	Exception Path
Receive Requests	RequestServer receives requests from invalid FieldUser.
Process Requests	The original request is not completed or incorrect for translation.
Send Requests	The request cannot be sent out since the Infrastructure is temporarily unreachable.
Receive Result	None.
Process Results	A "Not Found" result is received
Send Results	The result cannot be sent back to caller since the connection was broken.
Manage FW	Errors occur while executing the task.

2.4.2.4 Use-case Diagram of the Field Wrapper

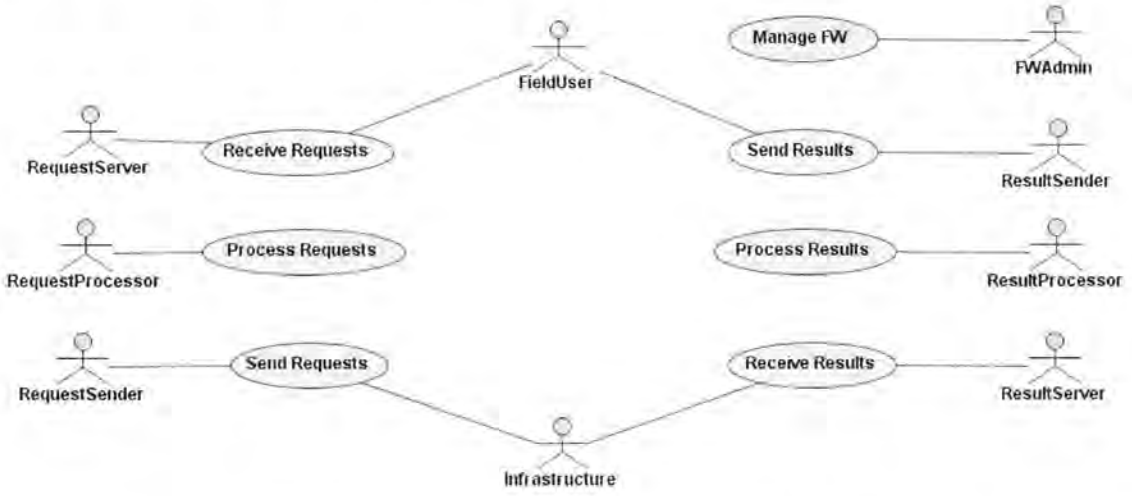

Figure 2-4. Use-case Diagram of the Field Wrapper

2.4.2.5 Identify Classes of the Field Wrapper

The classes are identified as listed as follows.

- **Entity Classes**
  - FieldUser
  - RequestServer
  - RequestProcessor
  - RequestSender
  - ResultServer
  - ResultProcessor
  - ResultSender
  - Request
  - Result
  - Format
  - ID
  - RequestList

- ConnectedUserList
- Report
- Setting
- Task

- **Boundary Classes**
  - ManageFWPanel
- **Control Classes**
  - Process Request Controller
  - Process Result Controller

2.4.2.6 Identify relationships between classes

The relationships between the classes are shown in table 2-7.

Table 2-7. Association and Multiplicity

Class		Association		Class
FieldUser	1	Submits	0..*	Request
FieldUser	1	Contains	1	RequestList
FieldUser	1	Contains	1	ID
RequestServer	1	Receives	0..*	Request
RequestProcessor	1	Processes	1	Request
RequestSender	1	Sends	1	Request
RequestProcessor	1	Changes	1	Format
ResultProcessor	1	Changes	1	Format
ResultProcessor	1	Modifies	1	Request
RequestProcessor	1	Modifies	1	ConnectedUserList
RequestServer	1	Modifies	1	RequestList
RequestSender	1	Modifies	1	RequestList
ResultSender	1	Modifies	1	Request
ResultServer	1	Receives	0..*	Result
ResultProcessor	1	Processes	1	Result
ResultSender	1	Sends	1	Result
FWAdmin	1	Requests	0..*	Report
FWAdmin	1	Changes	0..*	Setting
FWAdmin	1	Performs	0..*	Task

2.4.2.7 Class Diagram of the Field Wrapper

The class diagram of the field wrapper is shown in Figure 2-5.

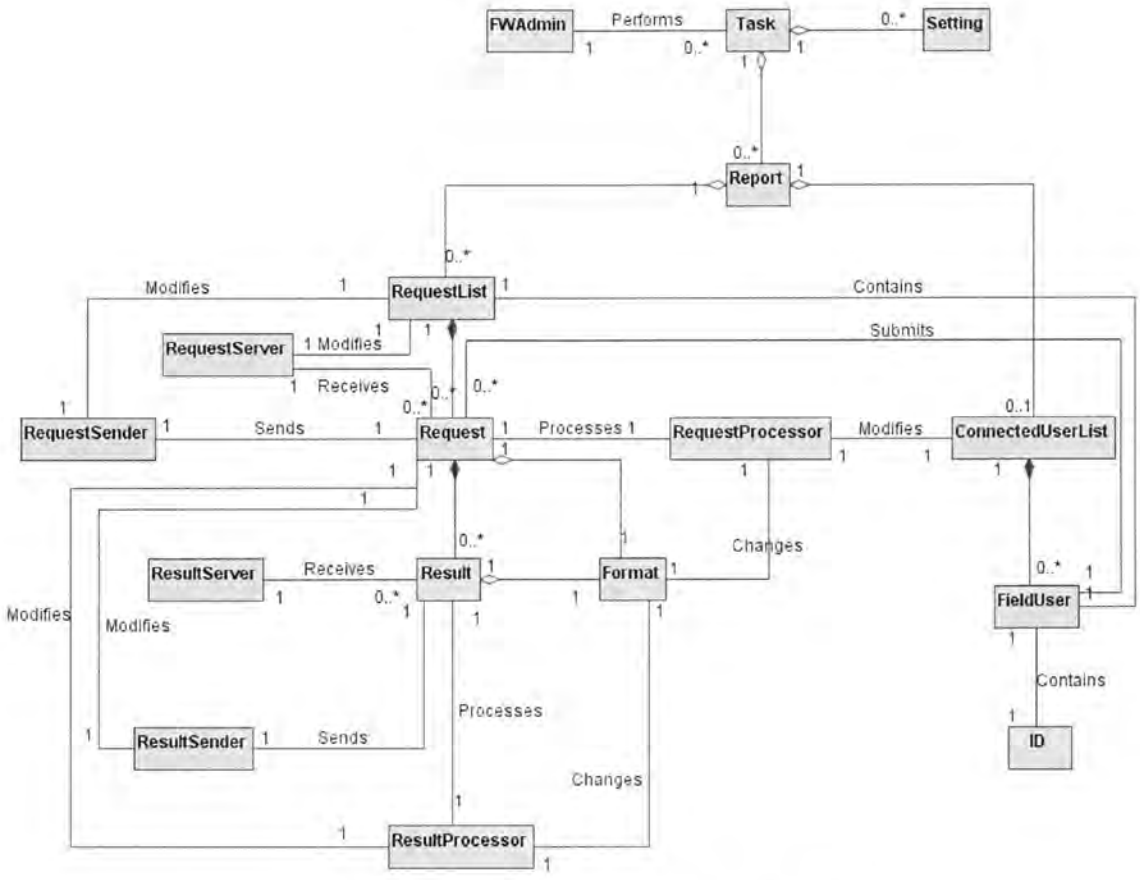

Figure 2-5. Class Diagram of the Field Wrapper

Chapter 3 Implementation of the Field Wrapper

3.1 Implementation Approaches

Several approaches must be considered in the implementation stage.

- **Use SOAP to address the firewall issue**

  The introduction of SOAP technology and implementation will be discussed in details in the next section.

- **Use a hash table to manage connected users**

  Every time a user wants to the use the system, he/she shall log in first using his user id and password. The field wrapper will contact the registration node to verify the user authentication information.

  If the login information the user provides is correct, a unique key will be calculated and saved in a hash table as a reference to that particular user. The corresponding user info, application info and device info will be saved in the hash table as the value. The unique key will be sent back to the field user along with the confirmation message of successful login. Once the user logs in successfully, every time a request is sent to the field wrapper will contain the unique key instead of sending the information of the user, device and application again.

  The efficiency of the field wrapper will be enhanced by this means since the field wrapper only needs to parse the user, device and application info once.

- **Add the information of the user/device/app to the request**

  The field wrapper shall be able to combine the information about the user/device/application with the requests sent from the user and generate a new request to be sent to the infrastructure.

- **Add a console to the field wrapper**

A (GUI) console will be handy for an administrator or a tester to manage, test or modify the field wrapper. The administrator or operator shall be able to view and set most of the properties of the field wrapper through the console.

- **Make use of MVC design pattern**

  Model-View-Controller (MVC) is a widely used software design pattern that was created by Xerox PARC for Smalltalk-80 in the 1980s. MVC design pattern enforces the separation between the input, processing, and output of an application. To this end, an application is divided into three core components: the model, the view, and the controller [13]. Each of these components handles a discreet set of tasks. Therefore, the code base is better organized and the code reusability and maintainability is improved.

3.2 Structure of the field wrapper

With the UML Design and all the analysis, the structure of the field wrapper is shown in Figure 3-1.

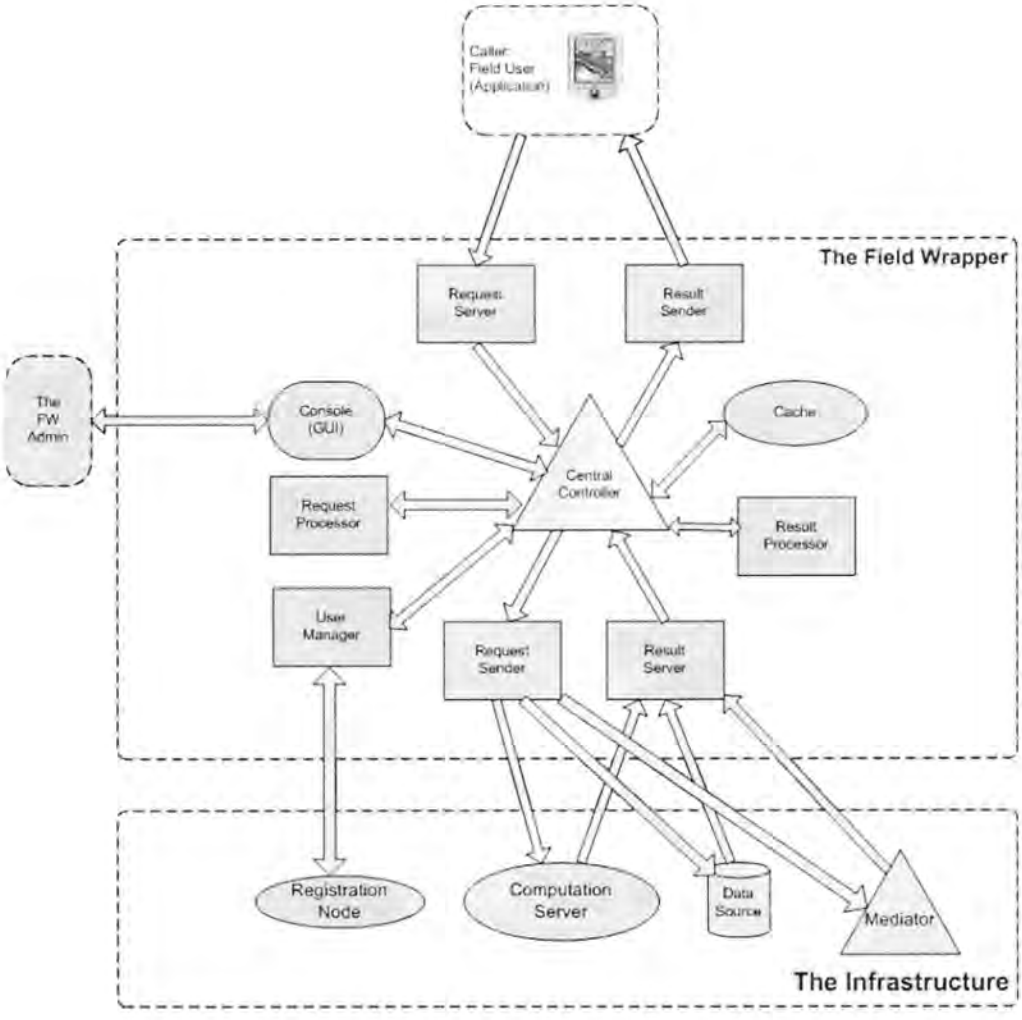

Figure 3-1. Structure of the Field Wrapper

3.2.1 Description of the components

- **Central Controller**

  Central Controller is the key controller of the whole field wrapper while applying the MVC design pattern. It is the only component that could communicate with all the other components directly. In another word, the central controller is the manager of the field wrapper, which controls the other components and makes them work as a single unit.

The central controller will initialize the other components at the beginning of execution. It will prompt error messages if it fails to initialize any of them. While exiting the field wrapper, the central controller will check whether it is safe to exit and release the corresponding memory allocation.

- **Request Server**

  Request Server is the connector between the field user and the field wrapper. It keeps listening to a specific port for incoming requests and passes the request to the central controller.

- **Request Processor**

  Request Processor is the component that parses and translates the requests from the field user. It shall be able to parse the request to check which part of the infrastructure it shall send the request to. The Request Parser and Request Combiner are two components contained in the Request Processor to parse and generate the request to be sent out.

- **Request Sender**

  Request Sender is the component that uses appropriate method to send the request to the destination. The connection method could be agent based or SOAP, etc.

- **(GUI) Console**

  (GUI) Console is an administration console for the field wrapper. It makes the field wrapper easy to be managed and it is good for demo and testing purpose as well. It could display every event happens in the field wrapper such as the arrival of a new request or a returned result, an error parsing request, etc..

- **Result Server**

  Result Server is the connector between the field wrapper and the infrastructure. It keeps listening to a specific port for incoming results and passes the result to the central controller.

- **Result Processor**

After receiving a result, a new Result Processor thread will be created to parse the result and deliver the parsed result to Result Sender to be sent to the corresponding field user. For example, in a moving field user case, the Result Processor will cache the image sequence and send the needed image to the Result Sender periodically.

- **Result Sender**

  Result Sender is the component that uses appropriate method to send the result back to the corresponding field user. The connection method could be socket, agent or etc..

- **User Manager**

  User Manager is the component that is in charge of verifying the identification of the user and manages the information of the currently connected user, device and application. It shall be able to communicate with the Registration Node in the infrastructure to verify whether he is a legal user. And for the efficiency of the field wrapper, it will store the information of the currently connected user in a hash table. The User Manager will also play an important role in mobile collaboration.

3.2.2 The Working Mechanism of the Field Wrapper

1. **Field Wrapper initialization**

   While starting the field wrapper, the central controller will be instantiated, and the other components will be initialized by the central controller. Also a login message will be sent to the infrastructure to register itself. If every component has been created successfully, the GUI will display a message and start listening for the coming request.

2. **User Login (Registration)**

   If a user wants to use the application, he should log in first and the application will send his user id and password together with the properties of the device and the application. The socket server will catch the request and pass the object (request) to the request processor via the central controller. The central controller will pass the information to

the user manager then the user manager will communicate with the registration node to verify it. If succeessful, the user manager will add the user info into the hash table and the socket server will return a true value to the application.

3. **Process a request (Request Translation)**

   If the user sends in a request, the Request Server will catch it then the central controller will invoke a new Request Processor thread to parse the request, which is the simple mediation. The processor will not only find the destination of the request, but also combine the stored user information from the user manager to the request, which forms the final request that will be sent from the field wrapper.

4. **Send a request out**

   The modified request will be passed to the Request Sender. The Request Sender will send the request to the desired destination, which could be the Mediator, Computation Server or specific data sources. If the destination is not reachable at that time, an error message will be generated in the console and also be sent back to the field user.

5. **Parse and send back the returned result**

   The Result Server will catch any returned result and pass it to the controller. The destination information will be extracted from the returned result by the Result Processor and the Result Sender will send the result back to the user.

6. **User Logout**

   A logout request will be sent to the field wrapper when the user chooses to close the application. The request server will catch the request then the corresponding entry in the hash table of the currently connected user will be deleted, and finally a confirmation message will be returned to the user to indicate the end of the session.

7. **Stop the field wrapper**

   When the administrator chooses to stop the field wrapper, the central controller will generate a summary report and delete all the components to release the memory allocation.

3.3 Integrating SOAP with the Infrastructure

In order to address the firewall and security issue, the SOAP technology is leveraged and integrated with the infrastructure.

3.3.1 Introduction to SOAP Technology

SOAP (Simple Object Access Protocol) is a lightweight protocol for exchange of information in a decentralized, distributed environment. SOAP depends on the idea that any two computers on the Internet can communicate using HTTP, the protocol that powers the Web. (Actually, SOAP can be transmitted over nearly any high-level protocol, including SMTP and POP3, but HTTP is by far the most common.) It then transmits information using XML, the markup language that allows us to create tags and document standards. The server turns the incoming XML into an object method call, and then turns the object's response into an XML document that is returned as the HTTP response. Since both HTTP and XML are open standards, published by the World Wide Web Consortium, they can be (and are) implemented on a variety of platforms and, thus, interact without any trouble.

SOAP, as its name implies, expects to work with objects rather than simple procedure calls. Thus, SOAP client invokes a method on a particular object on the server. The method is specified in the body of the XML document itself, while the object with which it is associated is named in an HTTP "SOAPAction" header.

The server itself, including its name and the port number on which the SOAP request is transmitted, is known as the SOAP proxy. This makes sense when you consider that the HTTP server is simply relaying an object method invocation and isn't doing any of this work by itself.

The body of a SOAP request or response will be in XML. Each SOAP message--a request or response--consists of an optional SOAP header and a mandatory SOAP body wrapped inside of a SOAP envelope. The envelope identifies the contents as belonging to SOAP and sets out the namespaces that will be used for the rest of the message. The headers

describe the data in the body, and the body contains the method call or its results. The structure of a SOAP message is shown in Figure 3-2.

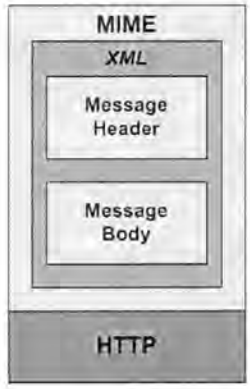

Figure 3-2. Structure of a SOAP message

In order to invoke an object on a remote server via SOAP, we will have to open an HTTP connection to the appropriate URL, identifying the object via the SOAPAction header. We send an XML document containing a SOAP envelope, inside of which the SOAP headers and body identify the method to be invoked on this object, as well as any parameters that the method might require. The client must additionally be prepared to parse the response returned by the SOAP server, extracting data structures contained in that response and using them as necessary.

3.3.2 SOAP Solution to Firewall Issue

To best protect the internal resources, network security personnel limit outside access as much as possible. Unnecessary services are deactivated and required services are placed behind firewalls that provide a single point of entry for external access to internal corporate applications, such as FTP servers, HTTP servers, and directory services. Incoming Internet traffic is inspected for required sources, destinations, traffic type, protocol, or other parameters.

Firewalls constrain programmer ability—they limit the exact behavior programmers are charged to create. Traditional remote access architectures, such as Remote Procedure Call

(RPC), the Common Object Request Broker Architecture (CORBA), and Remote Method Invocation (RMI), require multiple ports to be open through firewalls. In some systems, entire port ranges or protocols must be allowed through firewalls to allow applications to work correctly as shown in Figure 3-3 [6].

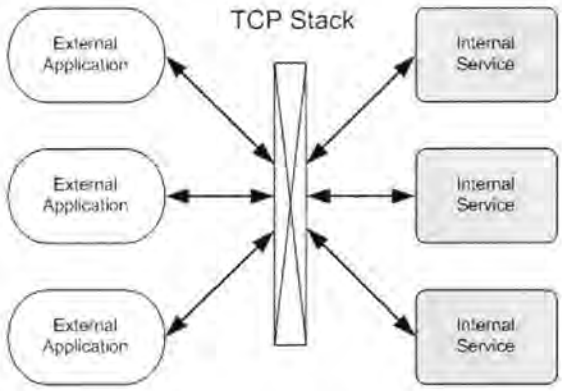

Figure 3-3. Traditional Model: Switching via Port

To enable sophisticated connections to remote personnel and outside organizations, programmers must be allowed some degree of freedom to connect services to each other. While most programmers understand the need for security, firewalls limit and add complexity to their applications.

SOAP is much simpler than traditional object communication technologies. And most importantly from the functionality perspective—it requires no additional ports or access beyond standard Web servers that exist in almost every organization. Since Web servers play integral roles in most corporate strategies, serving an organization's home page, for example, programmers can feel relatively secure that Web servers will not be firewalled. In effect, SOAP can tunnel through firewalls without any concessions from network security personnel as shown in Figure 3-4.

SOAP transmits messages by placing a routing engine (part of the SOAP processor) behind the Web server. Requests are sent through the Web server to the processor, where they are routed to the appropriate internal application. Responses are sent back through the

Web server to the remote client. Most firewalls inspect packets moving through an organization's Internet gateway. Once packets pass through the firewall, they enter a trusted domain and route to internal application servers. SOAP traffic is normally trusted because existing firewalls see it as "Web page requests" rather than powerful application messages. Once packets pass through the firewall, the SOAP processor unpacks embedded messages and routes them to internal application servers. The figure depicts differences between traditional message routing and SOAP message routing.

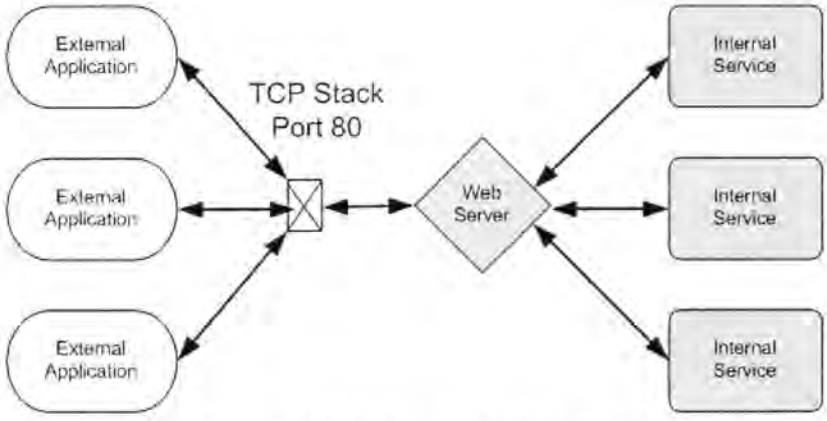

Figure 3-4. The SOAP Solution: Switching via Web Server

3.3.3 SOAP Implementation

In order to integrate SOAP with the existing infrastructure, while considering the field wrapper, the solution will be as shown in Figure 3-5. The field wrapper is the SOAP client in this case. And the SOAP server could be a part of the mediator, a plug-in to the mediator or a "wrapper".

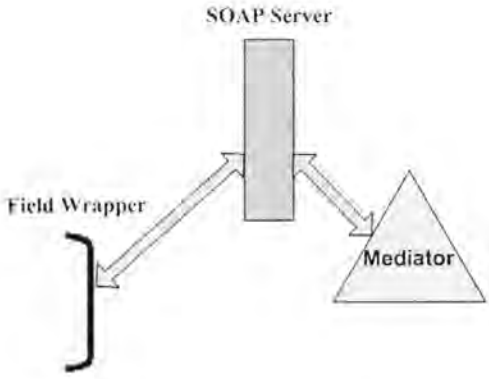

Figure 3-5. Integrate SOAP with the infrastructure

For simplicity purpose, the SOAP server was implemented as a wrapper to the mediator. As the SOAP client, the field wrapper will send the reconstructed requests to the SOAP server via HTTP protocol, the SOAP server will route the request to the mediator. Since basically a SOAP response will be sent back through the HTTP Server, but in the current infrastructure design, the computation server will send back the result to the field wrapper directly; we have to modify the existing mechanism to accommodate SOAP.

There are two ways to achieve that goal. One is letting the computation server send the result to mediator first, then the mediator passes the result back to the SOAP server to be sent back as a HTTP response to the previous HTTP (SOAP) request. Another way to implement it is more flexible and maintainable, while the SOAP server receives the request from the field wrapper, it routes the request to the mediator and sends a response immediately to the field wrapper as a confirmation. After the computation server gets the result, the computation server will send the result directly to the field wrapper's result server. The connection between the computation server and the field wrapper could be SOAP as well if firewall and security is taken into account, or some other connection methods such as agent, CORBA, etc..

Although the latter one is a better solution, the first method was implemented because the current infrastructure is relatively simple and it is relatively easier to be implemented.

Apache axis was chosen as the SOAP engine to create SOAP client and server. Apache Axis is one of the most popular Web services toolkits out there, which supports both RPC and Document-style services [7]. And Apache Tomcat [20] was chosen as the web server on which the axis engine was installed.

3.4 Field Wrapper Auto Generation

3.4.1 Needs in Auto Generation

The computer hardware and software techniques are changing rapidly, we can't imagine which programming language we are going to use to implement which part of the system, or what kind of connection method we are going to use to deliver messages between different components of the system in the coming decade. After all, a good design of the system should be upgradeable and maintainable.

Regarding the current infrastructure, some parts of the infrastructure such as the mediator and computation server are relatively fixed. But some parts of the infrastructure, for example, the field wrapper and data wrapper, are supposed to change frequently. For the field wrapper, we might have different types of applications installed in all kinds of devices in the future to make use of all the data sources connected to the infrastructure and special functions provided by the computation server. In additionally, the connection types from the device to the field wrapper might vary. In order to restrict the usage of the field wrapper according to different types of users, and also to make the field wrapper running more efficiently, we could make the field wrapper auto-generable which would provide the user the only functions that the user are legally to have or the only functions that the user needs, and nothing more.

3.4.2 Current Code Generation Technologies

XDoclet [9] and Velocity [10] are two famous code generation engines nowadays.

Xdoclet is an open source tool that extends the Javadoc Doclet API, allowing for the creation of files based on Javadoc @ tags and template files (.xdt) [8]. Although XDoclet is claimed to be able to generate 85% of the code, the most popular use of XDoclet is to generate EJB files such as deployment descriptors, remote and home interfaces, and even vendor-specific deployment descriptors, but not for customized java source files.

In this case, Velocity will be a better tool for code generation of the field wrapper. Velocity is a Java-based template engine and is also an open source tool. It defines its own language, the Velocity Template Language (VTL), to provide the easiest, simplest, and cleanest way to incorporate dynamic content with the predefined template file, as shown in Figure 3-6 [11].

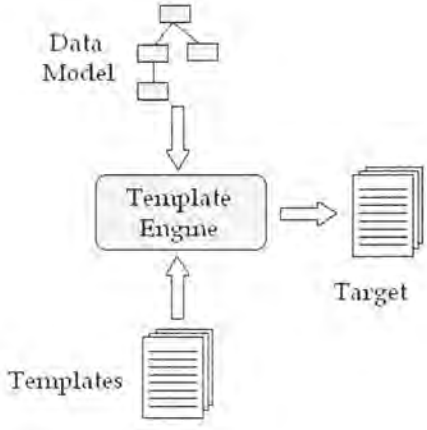

Figure 3-6. Template-based transformation

3.4.3 Auto Generation Approaches

The idea is to write Velocity templates based on the current field wrapper source code. Use VTL in the templates to add flow control to the code generation process according to the selection of the user, on which functionalities are required and which are not. Therefore, a java program is needed to read the selections of the user from the web page, read the templates from the hard drive and write the generated files back to the disk. In this case, an automation Ant script [12] will be extremely helpful. By using Apache Ant, the main java

program will be launched first, after the generated source files have been written to the disk, Ant could compile the source files, package the class files into one jar file and finally send the jar file to the user via email.

Chapter 4 Issues in Mobile Collaboration

4.1 Needs in Collaboration Using Mobile Computers

The current infrastructure is designed to be responsible for search, return and manipulate of geo-spatial data from all connected data sources according to the requests of the users. For the field user, that's one way to obtain information. But in order to accomplish his/her task, it is far from enough. Most of the time, the field user is not an expert, he/she might encounter with all kinds of problems as follows:

- The user has difficulties in interpreting the returned geo-spatial data such as charts and maps.
- The user needs his/her supervisor to make the decision.
- The user needs advice/help from people who is more familiar with the area/topic he/she is working on.

As presented above, to resolve the problems, the user not only needs the information (geo-spatial data) from the data sources, but also needs the information from the other people, in other words, collaboration is needed.

Let's consider the following scenario segment of collaboration:

David decides to interact with a colleague that is well versed in the area of South Dakota in question and review some additional information. He connects to a whiteboard to make anything that he looked at on his IPAQ available on his colleague's screen. To start the connection, he posts the original messages, his interpretation of the messages and the region map to his colleague. Fortunately, David's colleague is available and also signs on to the whiteboard.

While waiting for his colleague to look through the initial material, David starts to wonder about what kind of history anthrax had in the area in question. To get an idea, he requested a map showing the breakdown of the area with respect to anthrax infections over

the last 100 years. As he was looking at the map, he received a notice that his colleague had acknowledged the post and had pulled up a map showing the recent rainfall in the region of concern. David opened the rainfall map and was struck by how dry the area had been. After a quick exchange of messages, they ...

From the above scenario, we could see that collaboration is a very common but important idea in the real world. In the past, there were also some unavoidable issues while doing collaborations. For example:

- The user doesn't know from whom to get help.
- The user could not find the specific person. (or the user spend a lot of time in doing so)
- The situation is hard to explain verbally or by text. For example, map or the environment.

Nowadays, with the development of mobile computing and current infrastructure, suppose we are able to solve these issues. The infrastructure can integrate with a database with the list of experts for the user to select and contact with. The infrastructure can also push the request to the expert's windows once he/she connects to the system. With the build-in digital camera and the idea of "whiteboard", the expert can see what the user sees as if he/she is with the user physically.

4.2 Collaboration Basis

Collaborative platforms are generally characterized in terms of the set of concepts they support. The most basic of them is the concept of a collaborative session. A collaborative session is an activity of a group of people, the virtual team, which exchanges information among members. Collaborative sessions can be roughly classified on the basis of the dynamism of the information exchange in two categories: synchronous and asynchronous. A high level of interaction among the team characterizes synchronous sessions: all users share a single view of the discussion and information is exchanged as soon as it becomes available.

Conversely, in asynchronous sessions, information is transferred only on demand, resulting in a lower degree of interaction among the team members.

The architectures of collaborative platforms can be roughly classified into centralized and replicated [16]. In a centralized architecture, the shared application is maintained in a single physical location, and users are supplied with the output of this single application. In a replicated architecture, each user owns an instance of the application and the platform provides the mechanisms for synchronizing the various instances in order for all users to have a coherent view.

4.3 Possible Cases in Mobile Collaboration

Ideally, the collaboration could happen between two people, between a group and a person or among multiple people. Since every user, including the experts, connect to the infrastructure through the Field Wrapper, every time a user connects to the infrastructure via different field wrapper, the User Manager of the field wrapper will take care of this and have the previous messages route via this new field wrapper to the user. Therefore not only the userid is needed, but also the fwid (the id of the Field Wrapper) is required for the infrastructure to locate a user.

Case 1: The Collaboration happens between two people

The collaboration happens between two specific people, for example, a user and an expert. The scenario will be as follows: When the user, say A, encounters with some problem, and he knows that somebody, say B, might be able to have the knowledge about it. Of course, A knows the userid of B. A chooses the collaboration type as "Collaborate with a specified person", then A inputs B's userid, the title and the content of the question. The question may contain some attachments, which could be a recorded audio/video file, some map/data stored in the application or some other file. The request will be send to the mediator for the first time along with the unique request id, which consists of the fwid, A's userid, B's userid, etc. Such a request will be placed under B's name in the registration node.

If B is connecting to the infrastructure at the same time, or once B logs in the system later, he will receive the request immediately. After understanding the problem, B replies A with a confirmation and from then on, a peer-to-peer connection is set up between A and B (between the two Field Wrapper actually). After receiving the confirmation from B, A now can start collaborating with B by launching a "Whiteboard", audio chatting or text chatting like ICQ. After some time, if A is satisfied with the solution that B provides, A could mark the request as "Finished" then a new message will be sent to the Mediator to update the status of the request. A can send the solution together so that the request along with the solution might be stored in the database for a later reference. B will also be informed then he could be able to archive the information (request and solution) locally or remove it from the list directly. A can do the same thing as well. After B receives the "Finished" message, the connection between A and B could be terminated. If A is not satisfied with the solution, A can choose another person or another collaboration type to continue working on the same request.

Case 2: The Collaboration happens between one person and a virtual group

The collaboration request is sent to a group of people firstly, and the collaboration takes place between two people eventually. A chooses the collaboration type as "Collaborate with a virtual group". Then A inputs the id of the group, the title and the content of the question and also some attachments. All the members of the virtual group shared the same message board. The request will be sent to the Mediator first and be posted in the message board of the virtual group. Every member logged in the system can see every new requests posted in the message board. A member of the virtual group, say B, browses the description of A's request in the message board and believes that he could solve the problem. Then B checks out the request from the message board and sends a confirmation message to A to set up a peer-to-peer connection to A. Once a request is checked out by some member, a special icon will be attached to the icon and nobody else could check it out again until B checks it back in. (That would work like a read/write lock.) All the other things are similar to case 1.

Case 3: The collaboration happens among multiple people.

This case is more like a discussion group. While there is no longer a peer-to-peer connection between users, a "Whiteboard" could be shared among more than 2 people at the same time which means a special collaboration tool must work as a controller to synchronize messages.

4.4 Implementation Suggestions

With the support of the existing Infrastructure, it will be much easier to bring collaboration with geospatial data to mobile computing environment. Several more components could be added to the existing infrastructure to bring in collaboration functionality.

In the current infrastructure, the Computation Server is the most powerful component with computational capability; also all the query results will be sent back to the Computation Server and saved in the active cache. Therefore, the collaboration system could be plugged in as one of the tools in the Computation Server's tool set. The collaboration system could run in a separated server but it should have a direct interface with the Computation Server.

The collaboration system should make use of the mediation capability of the Mediator to distribute the messages to appropriate user(s).

In order to support local collaboration or local group collaboration, a light-weigh plug-in could be developed for the Field Wrapper to support collaboration without interfacing with the infrastructure.

Chapter 5 Conclusions and Future Work

5.1 Summary

In order to integrate heterogeneous geospatial data resources to field mobile computing environment, such as survey and other types of data collection, in our research, an infrastructure is designed and implemented. Additionally, to integrate mobile computing with the fixed network, field wrapper is introduced to the infrastructure. In order to address the current needs in field mobile computing and unsolved issues, for example, the firewall and security issue, some new technologies are leveraged and applied such as SOAP, MVC design pattern, code generation, etc. Mobile collaboration is also discussed along with implementation suggestions.

5.2 Future Work

1. Mobile Collaboration

Mobile collaboration is a hot topic nowadays. With the infrastructure, which integrates all kinds of geospatial databases, it becomes much easier to integrate digital geospatial data with mobile collaboration. In order to support mobile collaboration, more components should be designed and implemented which add the collaboration support to the field wrapper, mediator and computation server. Since the computation server is very powerful in computing and all the request result will all be sent back it, it will be convenient to make the main collaboration tool a part of the computation server.

Also both the synchronous and asynchronous mode of collaboration should be discussed, and centralized collaborative platform is also worth thinking about.

2. Code Generation

All the wrappers in the infrastructure, including field wrapper and data wrapper are suppose to have different settings in order to wrap different mobile computing environment

or different data sources. Also "light weight" is another important characteristic of the wrappers, we don't want to the wrappers to perform some extra work which is not needed in its specific environment. Therefore, code generation is very important. We should keep working on the auto generation of the field wrapper and start looking at the auto generation of the data wrapper.

3. Leverage new XML security technologies

In addition to SOAP, while applying web service, some new XML security technologies such as W3C XML-Signature syntax [17], Security assertion markup language (SAML) [18], extensible access control markup language (XACML) [19] should be discussed and leveraged.

References

[1] L.L. Miller, Nikhil Sathe, Ming Hua, Dhigha Sekaran, Sarah Nusser and Pheishang Zhao, "An Infrastructure for Delivering Geospatial Data to Field Users", *The IASTED International Conference on Parallel and Distributed Computing and Systems.* Anaheim, CA. 2001.

[2] Sarah Nusser, Leslie Miller, Keith Clarke and Michael Goodchild, "Future Views of Field Data Collection in Statistical Surveys", *Proceedings of dg.o 2001 National Conference on Digital Government Research,* Los Angeles, CA. 2001.

[3] K. C. Clarke, A. Nuernberger, T. Pingel and D. Qingyun, "User Interface Design for a Wearable Field Computer", *Proceedings of dg.o 2002 National Conference on Digital Government Research,* 343-346, 2002.

[4] Upkar Varshney, "Networking Support for Mobile Computing", *Communications of the Association for Information Systems,* Volume 1, Article 1, 1999.

[5] Manuel Alba and Jesus Favela, "Supporting Handheld Collaboration through COMAL", *proceedings of the 6th International Workshop on Groupware,* 52-59, 2000.

[6] Conan C. Albrecht, "How Clean is the Future of SOAP?", *Communications of ACM,* Vol. 47, No. 2, 2004.

[7] Kevin Gibbs, Brain D. Goodman and Elias Torres, "Create Web Service Using Apache Axis and Castor", *IBM DeveloperWorks,* 30 September 2003.

[8] Eric M. Burke and Brain M. Coyner, "XDoclet Introduction", *Java Extreme Programming Cookbook,* O'Reilly, 16 November 2003.

[9] XDoclet Team, "Welcome! What is XDoclet", http://xdoclet.sourceforge.net/xdoclet/, 1 March 2003.

[10] Velocity Team, "The Velocity User Guide", http://jakarta.apache.org/velocity/user-guide.html, 16 November 2004.

[11] Giuseppe Naccarato, "Template-Based Code Generation with Apache Velocity", http://www.onjava.com, 05 May 2004.

[12] Eugene Kuleshov and Dmitry Platonoff, "Java Software Automation with Jakarta Ant", http://www.onjava.com, 24 July 2002.

[13] Brain Kotek, "MVC Design Pattern brings about better organization and code reuse", http://builder.com.com, 30 October 2002.

[14] D. Box. "Simple Object Access Protocol (SOAP) 1.1", World Wide Web Consortium (W3C), http://www.w3.org/TR/SOAP , 8 May 2000.

[15] Federico Bergenti, Agostino Poggi and Matteo Somacher, "A Colaborative Platform for Fixed and Mobile Networks", *COMMUNICATIONS OF THE ACM,* Vol. 45, No. 11, 2002.

[16] Minenko, W. "The Application sharing technology", *The X Advisor,* 1 January 1995.

[17] Ernesto Damiani, Sabrina De Capitani di Vimercati and Pierangela Samarati, "Towards Securing XML Web Services", *ACM Workshop on XML Security,* 15 November 2004.

[18] Prateek Mishra and Rob Philpott, "Advancing SAML, an XML-based security standard for exchanging authentication and authorization information", http://www.oasis-open.org/committees/security/, 15 November 2004.

[19] Hal Lockhart and Bill Parducci, "Defining XACML, an XML specification for expressing policies for information access over the Internet", http://www.oasis-open.org/committees/xacml, 15 November 2004.

[20] Apache Jakarta Tomcat Team, "Tomcat User Guide", http://jakarta.apache.org/tomcat/index.html, 15 November 2004.